SemEval 2022: Task 4, Subtask 1

CSCI 5832: Fall 2021 Shared Task

Nitin Kumar, Rohan Das, Tushar Gautam

Abstract

This report describes the system modeled to solve Task 4: Patronizing and Condescending Language Detection at SemEval 2022. Very specifically the system aims to present a viable solution for Subtask 1: Binary Classification. We fine-tuned two pre-trained language models, RoBERTa [3] and XLNet [4], on this task and performed a range of experiments that involved dataset preprocessing and addressing the substantial dataset imbalance. Our best performing model was a fine-tuned RoBERTa architecture with weighted classes.

Introduction

Our task at hand is to perform binary classification if a given sentence belongs to patronizing and condescending language (PCL). The aim of this task is to identify PCL, and to categorize the linguistic techniques used to express it, specifically in news stories, when referring to vulnerable communities.

We built on the foundational work done by the authors of the Don't Patronize Me! [1] dataset. The authors have done great work in expounding on PCL and analyzing why it is crucial to identify the presence of PCL in online media to protect the vulnerable communities in our society. The authors compared various language models and reported the best results seen on BERT [2] based models. To extend the work, we performed fine-tuning experiments on two pre-trained Transformer based models, namely RoBERTa (roberta-base-cased) and XLNet (xlnet-base-cased).

Background

PCL dataset has more than 10,000 paragraphs annotated with labels in range 0-4 based on the extent of agreement among the annotators, with 0 and 4 being total agreement on a paragraph being no-PCL (negative) and PCL (positive) respectively while 1-3 represents partial agreement. For the binary classification task, authors proposed binning the labels by considering 0-1 as negative label (0) and 2-4 as positive label (1).

Furthermore, the dataset is significantly imbalanced with only 10% positive samples in the dataset. An interesting observation is that the PCL dataset has a paragraph length of only 145 words with 99 percentile, while the longest paragraph has 820 words.

System Overview

We primarily tried out fine-tuning two state of the art pre-trained language models on the given dataset: RoBERTa and XLNet.

1. **RoBERTa:** RoBERTa builds on BERT's language masking strategy, wherein the system learns to predict intentionally hidden sections of text within otherwise unannotated language examples. RoBERTa, which was implemented in PyTorch, modifies key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective, and training with much larger mini-batches and learning rates. This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better downstream task performance.
2. **XLNet:** XLNet is an auto-regressive language model which outputs the joint probability of a sequence of tokens based on the transformer architecture with recurrence. Its training objective calculates the probability of a word token conditioned on all permutations of word tokens in a sentence, as opposed to just those to the left or just those to the right of the target token. XLNet performs substantially better than BERT on many language tasks.

We fine-tuned both of these language models individually on the pre-processed PCL dataset and subsequently performed hyperparameter tuning.

The biggest challenge on this task was dealing with the imbalance in the dataset. We tried out three methods to deal with this: downsampling the majority class, upsampling the minority class and adding class weights while computing loss.

Experimental Setup

1. **Data Splits:** We used two different splits: the default train/test split provided by the organizers and a custom 75-15-10 train/dev/test split.
2. **Data Preprocessing:** For the data preprocessing step, we removed URLs, punctuations and extra whitespaces. Stray HTML tags in the text were also processed and removed. We additionally collapsed words tokens like "we 're" into "we're" and expanded all "n't" to "not". All misspelled words where characters occurred more than twice were fixed as well. Finally all of the text was converted to lowercase.
3. **External Tools and Libraries used:** We used the Hugging Face, Simple Transformers and PyTorch libraries for training the models. Standard Python libraries like Scikit-Learn, Pandas, NLTK and BeautifulSoup were used for processing the dataset. Google Colab was used for fine-tuning the models.
4. **Evaluation Measures:** The competition's evaluation script was used to evaluate the model on an unseen test set. The model was evaluated on precision, recall and f1-score.

Setup for RoBERTa

Starting with the RoBERTA baseline discussed in the dataset paper, we explored different negative down-sampling ratios, and added weights to the labels while computing loss during training to tackle an imbalanced dataset. Without adding weights to the labels, we'd clearly see the model giving biased results (lower precision, higher recall) toward the majority label and not fully generalising the true representation of the dataset.

Furthermore, we compared results with negative downsampling ratios of 1:2, 1:5, 1:6, 1:7, 1:8 and no downsampling. Best results were seen for the non downsampling case. A possible explanation for this anomaly could be that since the given dataset is small, downsampling to an arbitrary ratio might have an adverse impact as the model might not fully generalise on the true distribution of the dataset. This could also be inferred with lower precision value for 1:2 or 1:5 negative downsampling compared to 1:7 or with no downsampling. On the contrary, the recall for 1:2 negative downsampling is much higher than what we observed with reduced downsampling. That's an indication of the model being biased and not learning the true representation of the dataset.

Setup for XLNet

XLNet requires specifically formatted inputs. For each tokenized input sentence, we needed to create:

1. **Input Ids**: a sequence of integers identifying each input token to its index number in the XLNet tokenizer vocabulary
2. **Attention Mask**: a sequence of 1s and 0s, with 1s for all input tokens and 0s for all padding tokens
3. **Labels:** a single value of 1 or 0. In our task 1 means "PCL" and 0 means "No PCL"

The model is fine-tuned on the preprocessed and tokenized training set over a range of hyperparameters[1]. Like RoBERTa, we experimented with different weights on the classes while computing the loss to tackle the imbalance in the dataset. We did not downsample the majority class for these experiments based on our unfavourable findings of downsampling when fine-tuning on RoBERTa. We instead chose to upsample the minority class with a ratio of 1:2 and 1:4, but saw no substantial improvement in the evaluation metrics. Recall seems to take a substantial hit compared to the non upsampling case.

Results:

RoBERTa

Even though RoBERTa's maximum sequence length is 512, we could not run experiments with 512 as maximum sequence length for roberta-base-cased model due to lack of computing

[1] Appendix - Table of Hyperparameters

resources. It is likely that this might not be a big issue since the 99th percentile paragraph length in the PCL dataset is only 145. So choosing 256 instead of 512 for *max_seq_length* might not hamper the results much. Furthermore, we saw poor performance for lower batch_size (8, 16) which can be explained by the fact that lower batch size would less likely have positive samples during the training phase. Furthermore, as briefly mentioned before, higher negative downsampling ratio performed poorly compared to lower negative downsampling or no downsampling, likely due to lack of sufficient data. So with just the weighted labels and no downsampling, we observe best results as reported in #10.

S.No.	#Epochs	Batch Size	Label Weights	max_seq_len	Downsampling	Precision	Recall	F1
1.	10	8	Balanced	128	1:2	0.34	0.71	0.46
2	10	32	Balanced	128	1:2	0.39	0.72	0.51
3	10	32	Balanced	256	1:2	0.40	0.76	0.53
4	15	32	Balanced	128	None	0.41	0.73	0.53
6.	10	32	[0.55, 5.27]	128	None	0.65	0.48	0.55
7.	10	32	[0.6, 3]	128	1:5	0.55	0.63	0.59
8.	10	32	[0.57, 4]	128	1:7	0.58	0.56	0.57
9.	10	64	[0.55, 5.27]	128	None	0.60	0.50	0.55
10.	**10**	**64**	**[0.55, 5.27]**	**256**	**None**	**0.69**	**0.54**	**0.61**

TABLE 1: Fine-Tuning results on RoBERTa

XLNet

We see our best results for XLNet are achieved with a maximum sequence length of 512, batch size of 8 and weighted classes. The batch size was constrained to 8 due to a lack of computing resources, so it would be interesting to see if we see better generalization for higher batch sizes of 32 and 64. We also noticed that performance substantially degraded when the model is fine-tuned for more than 4 epochs and the model tends to bias towards predicting the negative class more frequently, due to the class imbalance. Weighting the classes when computing the loss, to tackle the imbalance issue, showed promise for maximum sequence length of 512 but potentially better performance was restricted by the small batch size of 8.

S.No.	#Epochs	Batch Size	Label Weights	max_seq_len	Upsampling	Precision	Recall	F1
1.	2	32	Balanced	128	None	0.55	0.58	0.57

2.	4	32	Balanced	128	None	0.62	0.49	0.55
3.	4	48	Balanced	128	None	0.51	0.56	0.53
4.	4	16	Balanced	256	None	0.45	0.62	0.52
5	2	16	[0.6, 3]	512	None	0.50	0.58	0.53
6.	**4**	**8**	**[0.55, 5.27]**	**512**	**None**	**0.59**	**0.56**	**0.58**
7	4	16	[0.55, 5.27]	512	None	0.53	0.63	0.57
8.	4	8	Balanced	512	1:2	0.61	0.48	0.54

TABLE 2: Fine-Tuning results on XLNet

For both RoBERTa and XLNet we cannot conclusively say if data preprocessing had any substantial effect on the performance. A possible explanation could be that the dataset did not have a lot of noise that needed cleaning in the first place.

Model	**Precision**	**Recall**	**F1**
roberta-base-cased	**0.69**	**0.54**	**0.61**
xlnet-base-cased	0.59	0.56	0.58
roberta-baseline	0.35	0.78	0.48

TABLE 3: Cumulative Results

Conclusion

PCL detection is difficult both for humans and NLP systems, due to its subtle nature, its subjectivity and the fair amount of world knowledge and commonsense reasoning required to understand this kind of language. This coupled with the highly imbalanced nature of the dataset makes it difficult for even state of the art transformer models to inherently learn to identify PCL. Adding class weights when computing the loss on a fine-tuned RoBERTa architecture seemed to be the best way to tackle the imbalance compared to upsampling/downsampling the data and shows substantial improvement over the competition baseline.

One potential area of exploration could be to try and better model both word and sentence level context while trying to incorporate linguistic theories around what constitutes patronizing and condescending language. Another interesting avenue could be to explore anthropological cues around groups and communities targeted with PCL that could lead us to develop clustering algorithms around PCL specific keywords which can then be made a part of ensemble models.

References

[1] Perez-Almendros, Carla & Espinosa-Anke, Luis & Schockaert, Steven. (2020). Don't Patronize Me! An Annotated Dataset with Patronizing and Condescending Language towards Vulnerable Communities. 5891-5902. 10.18653/v1/2020.coling-main.518.

[2] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[3] Liu, Yinhan & Ott, Myle & Goyal, Naman & Du, Jingfei & Joshi, Mandar & Chen, Danqi & Levy, Omer & Lewis, Mike & Zettlemoyer, Luke & Stoyanov, Veselin. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.

[4] Yang, Zhilin & Dai, Zihang & Yang, Yiming & Carbonell, Jaime & Salakhutdinov, Ruslan & Le, Quoc. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding.

Appendix

Hyperparameters

Parameters	**Values**
batch_size	8, 16, 32, 64, 128
max_seq_len	128, 256, 512

label weight	(Proportional to downsampling ratio)
	[0.55, 5.27] (say for 1:10 ratio in original dataset)
	`w_pos = (total_sample)/(2*n_sample_pos)`
	`w_neg = (total_sample)/(2*n_sample_neg)`

warmup_steps	500
weight_decay	0.01
learning_rate	1e-05, 2e-05, 4e-05, 5e-05

TABLE 4: Hyperparameters